MA4832

Microprocessor Systems

Laboratory Session 1

(Debug, Demo, Push Button, Buzzer, LED, Dip Switch)

Overview

Lab1 includes the following practices:

1. to use debug menu commands to step through the example's codes.
2. to download and run a demo program in TM4C123G Launch Pad.
3. to read in and store a 4-bit word as specified by a dip switch.
4. to turn on LEDs.
5. to use a push button to activate a buzzer.

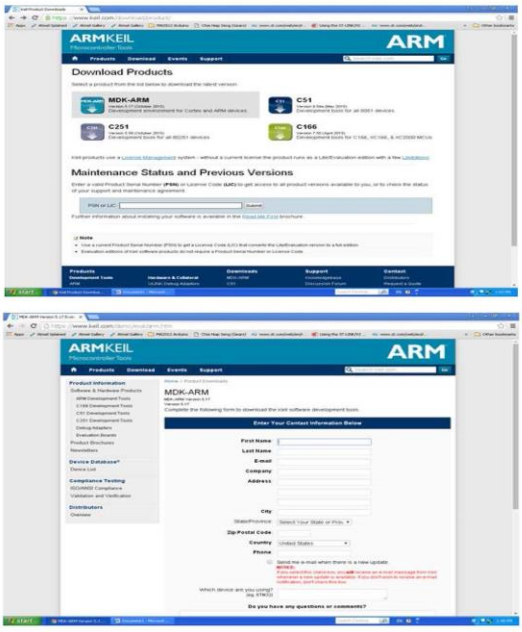
Practice 1: Step Through Sample Codes

Preparation for Personal Installation

- Download/install Keil uVision5 IDE
- Configure IDE for Tiva C board
    - EK-TM4C123GXL .

Download, install, configure IDE

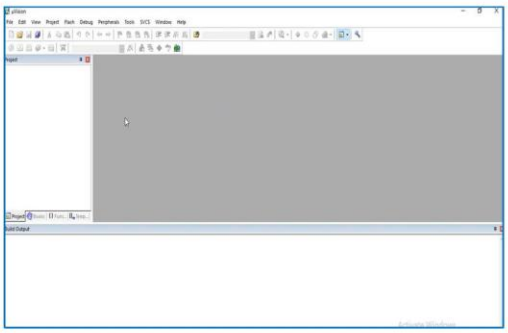
- Download from: https://www.keil.com/download/product/
- Fill out the form to request for free download of Lite/Evaluation edition



Create MA4832 Folder

- Create a folder "MA4832" on your desktop with sub folders
  - Lab 1, Lab 2, Lab 3, Lab 4
- Populate subfolder Lab 1 with the following subfolders:
  - Simulation
    - startup.s, prog1.s
  - with Hardware
- Watch the video: Media0.mp4
- Duplicate the steps, below.

All resources may be found on NTULearn

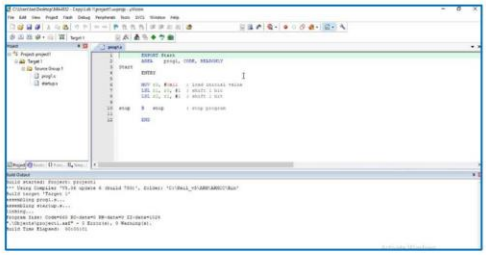


Create a New Assembly Project in Keil uVision5 IDE.

1. Launch program named "Keil uVision5".
2. Start creating assembly project.
   - Choose "New uVision Project" from Project menu.
   - Select working folder and enter a new project filename.
   - Select Device for Target, in the Data base tree, choose the vendor and then the chip you want to use and click OK.
     (For this lab, select the vendor, Texas Instruments and the chip, TM4C123GH6PM in "Tiva C Series -> TM4C123x Series").
   - Click "OK" to exit the Manage Run-Time Environment.
3. Add the existing source files to the project.
   - Expand Target 1 -> Right click Source Group 1 -> select Add Existing Files to Group 'Source Group 1'
   - Select Files of type Asm Source file (*.s*; *.src; *.a*)
   - Select two existing files, named "startup.s" and "prog1.s" which can be found in your working folder, "Lab 1" -> "Software simulation".
4. Build the program.
   - click on the Build icon or choose build target from the Project menu.
   - If the program is built successfully, the message "0 Error(s), 0 Warning(s)" will appear on the Build Output window at bottom of the screen.

Exploring Debugging Functions

- Click on link to Media1.mp4 to launch video
- Open Project containing Prog1.s
- Setup IDE as depicted in the figure
- Follow steps listed below.



Detail Processes

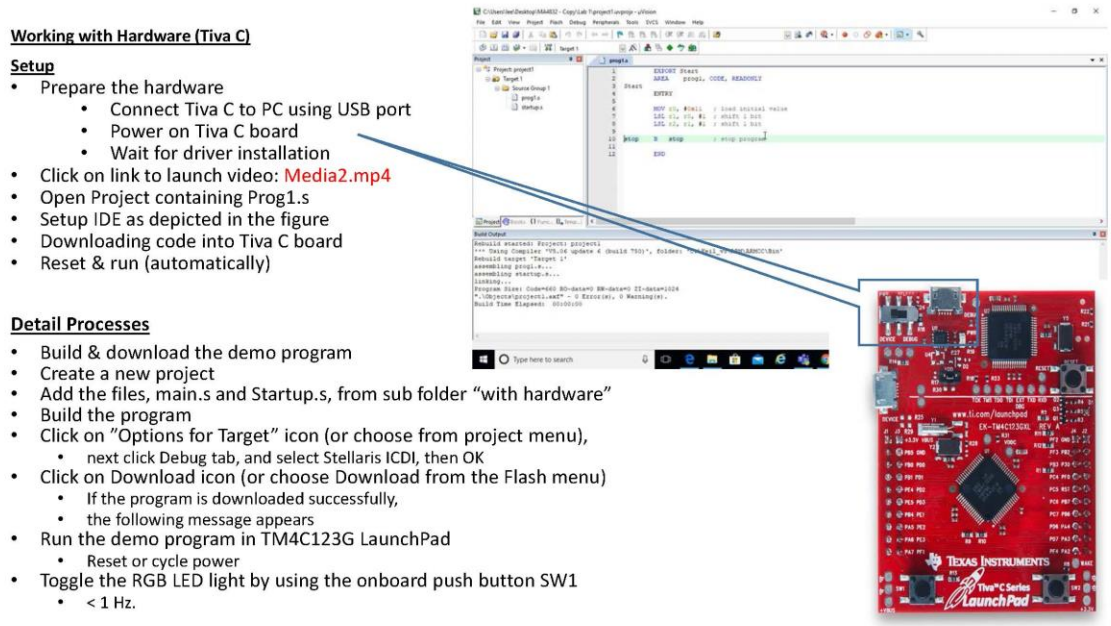
Click on "Options for Target" icon (or choose from project menu), next click Debug tab, and select "Use Simulator", then OK

- Click Start/Stop Debug Session icon or choose Start/Stop Debug Session from the Debug menu. (or simply press Ctrl+F5)
- Click OK for Evaluation mode
- If it starts tracing successfully, a cursor appears in front of the next instruction to be executed.
- Click the "Reset" icon and start to trace the program using the "Step Over" icon or click on "Step Over" from the Debug menu.
  It executes the instructions of the program one after another.
- To trace the program you can use the "Step" icon, as well.

Note:

- The difference between the Step Over and Step is in executing functions.
  - While Step goes into the function and executes its instructions one by one, Step Over executes the function completely and goes to the instruction next to the function.
  - To see the difference between them, trace the program once with Step Over and then with Step.
  - When you are in the function and you want the function to be executed completely you can use Step Out. In the case, the instructions of the function will be executed, it returns from the function, and goes to the instruction which is next to the function call.
- To restart the program, click the "Reset" icon to reset the CPU.
- To run the program without tracing, click the "Run" icon.
- To exit from the debugging mode, click "Start/Stop Debug Session" icon or choose "Start/Stop Debug Session" from Debug menu.

Practice 2: Run Demo Program on TM4C123G Launch Pad



Working with Hardware (Tiva C)

Setup

- Prepare the hardware
  - Connect Tiva C to PC using USB port
  - Power on Tiva C board
  - Wait for driver installation
- Click on link to launch video: Media2.mp4
- Open Project containing Prog1.s
- Setup IDE as depicted in the figure
- Downloading code into Tiva C board
- Reset & run (automatically)

Detail Processes

- Build & download the demo program
- Create a new project
- Add the files, main.s and Startup.s, from sub folder "with hardware"
- Build the program
- Click on "Options for Target" icon (or choose from project menu),
  - next click Debug tab, and select Stellaris ICDI, then OK
- Click on Download icon (or choose Download from the Flash menu)
  - If the program is downloaded successfully,
  - the following message appears
- Run the demo program in TM4C123G LaunchPad
  - Reset or cycle power
- Toggle the RGB LED light by using the onboard push button SW1
  - < 1 Hz.

Practice 3: Read in Bits (Input)

Task: to read info from Dip Switch.

You are given with:

- Schematic of the sample circuit is shown in Fig. 1.
- Program: rd_portA.s

```
; rd_portA.s
; read PortA Bit 4-7 (Pins PA4 - PA7 are connected to dip switch)

; GPIO_PORTA address

GPIO_PORTA_DATA_R      EQU 0x400043FC
GPIO_PORTA_DIR_R       EQU 0x40004400
GPIO_PORTA_AFSEL_R     EQU 0x40004420
GPIO_PORTA_PUR_R       EQU 0x40004510
GPIO_PORTA_DEN_R       EQU 0x4000451C
GPIO_PORTA_AMSEL_R     EQU 0x40004528
GPIO_PORTA_PCTL_R      EQU 0x4000452C
PA_4567                EQU 0x400043C0        ; PortA bit 4-7

SYSCTL_RCGCGPIO_R      EQU 0x400FE608        ; GPIO run mode clock gating control

               THUMB
```

```
                AREA    DATA, ALIGN=4
                EXPORT  Result [DATA,SIZE=4]
Result          SPACE   4

                AREA    |.text|, CODE, READONLY, ALIGN=2
                THUMB
                EXPORT  Start

Start

; initialize Port A
; enable digital I/O, ensure alt. functions off

; activate clock for PortA
                LDR R1, =SYSCTL_RCGCGPIO_R           ; R1 = address of SYSCTL_RCGCGPIO_R
                LDR R0, [R1]                         ;
                ORR R0, R0, #0x01                    ; turn on GPIOA clock
                STR R0, [R1]
                NOP                                  ; allow time for clock to finish
                NOP
                NOP

; no need to unlock Port A bits
; disable analog mode
                LDR R1, =GPIO_PORTA_AMSEL_R
                LDR R0, [R1]
                BIC R0, R0, #0xF0                    ; disable analog mode on PortA bit 4-7
                STR R0, [R1]

; configure as GPIO
                LDR R1, =GPIO_PORTA_PCTL_R
                LDR R0, [R1]
                BIC R0, R0,#0x00FF0000               ; clear PortA bit 4 & 5
                BIC R0, R0,#0XFF000000               ; clear PortA bit 6 & 7
                STR R0, [R1]

; set direction register
                LDR R1, =GPIO_PORTA_DIR_R
                LDR R0, [R1]
                BIC R0, R0, #0xF0             ; set PortA bit 4-7 input (0: input, 1: output)
                STR R0, [R1]

; disable alternate function
                LDR R1, =GPIO_PORTA_AFSEL_R
                LDR R0, [R1]
                BIC R0, R0, #0xF0             ; disable alternate function on PortA bit 4-7
                STR R0, [R1]

; pull-up resistors on switch pins
                LDR R1, =GPIO_PORTA_PUR_R            ;
                LDR R0, [R1]                         ;
                ORR R0, R0, #0xF0                    ; enable pull-up on PortA bit 4-7
                STR R0, [R1]

; enable digital port
                LDR R1, =GPIO_PORTA_DEN_R
                LDR R0, [R1]
                ORR R0, R0, #0xF0                    ; enable digital I/O on PortA bit 4-7
                STR R0, [R1]

                LDR R1, =PA_4567

Loop
                LDR R0, [R1]                         ; R0 = dip switch status
                LDR R2, =Result
                STR R0,[R2]                          ; store data

                B Loop

                ALIGN                          ; make sure the end of this section is aligned
                END                            ; end of file
```

- Load and run the program.
- Observe the following points:
  (i) How to read in a 4 bit word via GPIO Port A
  (ii) Check the value at address 0x20000000, it should give you the value indicated by the dip switch.

Practice 4: Turn on LEDs (Output)

Task: to output data which will turn on LEDs.

You are given with:
- Schematic of the sample circuit is shown in Fig. 2
- Program: wr_portB.s

```
; wr_portB.s
; output the value to PortB bit 0 - 3 (Pins PB0 - PB3 are connected to LEDs)

; GPIO_PORTB address

GPIO_PORTB_DATA_R      EQU 0x400053FC
GPIO_PORTB_DIR_R       EQU 0x40005400
GPIO_PORTB_AFSEL_R     EQU 0x40005420
GPIO_PORTB_PUR_R       EQU 0x40005510
GPIO_PORTB_DEN_R       EQU 0x4000551C
GPIO_PORTB_AMSEL_R     EQU 0x40005528
GPIO_PORTB_PCTL_R      EQU 0x4000552C
PB_0123                EQU 0x4000503C                ; Port B bit 0-3

SYSCTL_RCGCGPIO_R      EQU 0x400FE608                ; GPIO run mode clock gating control

        AREA    |.text|, CODE, READONLY, ALIGN=2
        THUMB
        EXPORT  Start

Start

; initialize Port B, all bits
; enable digital I/O, ensure alt. functions off

; activate clock for Port B
                LDR R1, =SYSCTL_RCGCGPIO_R            ; R1 = address of SYSCTL_RCGCGPIO_R
                LDR R0, [R1]
                ORR R0, R0, #0x02                     ; set bit 1 to turn on clock for GPIOB
                STR R0, [R1]
                NOP                                   ; allow time for clock to finish
                NOP
                NOP

; no need to unlock Port B bits
; disable analog mode
                LDR R1, =GPIO_PORTB_AMSEL_R
                LDR R0, [R1]
                BIC R0, R0, #0x0F                     ; Clear bit 0-3, disable analog function
                STR R0, [R1]
```

```
; configure as GPIO
                LDR R1, =GPIO_PORTB_PCTL_R
                LDR R0, [R1]
                BIC R0, R0,#0x000000FF              ; bit clear PortA bit 0 & 1
                BIC R0, R0,#0X0000FF00              ; bit clear PortA bit 2 & 3
                STR R0, [R1]

; set direction register
                LDR R1, =GPIO_PORTB_DIR_R
                LDR R0, [R1]
                ORR R0, R0, #0x0F             ; set PortB bit 0-3 as output (0: input, 1: output)
                STR R0, [R1]

; disable alternate function
                LDR R1, =GPIO_PORTB_AFSEL_R
                LDR R0, [R1]
                BIC R0, R0, #0x0F             ; disable alternate function on PortB bit 0-3
                STR R0, [R1]

; enable digital port
                LDR R1, =GPIO_PORTB_DEN_R
                LDR R0, [R1]
                ORR R0, #0x0F                       ; enable PortB digital I/O
                STR R0, [R1]

                LDR R1, =PB_0123

                LDR R0, =0x0F                       ; set PortB bit 0-3 -> turn on 4 Leds
                STR R0, [R1]
                BL Delay
                LDR R2,=0x0F
Loop
                EOR R0, R2                          ; R0 = Exclusive OR of R0 with R2
                STR R0, [R1]                        ; clear PortB bit 0-3 -> turn off 4 Leds
                BL Delay
                B Loop

Delay
                MOV R7,#0xFFFFFF

Countdown
                SUBS R7, #1                         ; subtract and sets the flags based on the
result
                BNE Countdown

                BX LR                               ; return

                ALIGN                               ; make sure the end of this section is
aligned
                END                                 ; end of file
```

- Load and run the program
- Observe the following points:
  (iii) How to output a 4 bit word via GPIO Port B
  (iv) Branching and looping

Practice 5: Interact with Button and Buzzer (Input and Output)

Task: To read in data from Push Button and to send out command to Buzzer.
You are given with:

- Schematic of the sample circuit is shown in Fig. 3
- Program: prog_portD.s

```
; prog_portD.s
; read portD bit 0 (push button) and output the value to portD bit 3 (buzzer)

; GPIO_PORTD address

GPIO_PORTD_DATA_R      EQU 0x400073FC
GPIO_PORTD_DIR_R       EQU 0x40007400
GPIO_PORTD_AFSEL_R     EQU 0x40007420
GPIO_PORTD_PUR_R       EQU 0x40007510
GPIO_PORTD_DEN_R       EQU 0x4000751C
GPIO_PORTD_AMSEL_R     EQU 0x40007528
GPIO_PORTD_PCTL_R      EQU 0x4000752C
PD                     EQU 0x40007024        ; Enable Port D bit 0 and 3

SYSCTL_RCGCGPIO_R      EQU 0x400FE608        ; GPIO run mode clock gating control

        AREA    |.text|, CODE, READONLY, ALIGN=2
        THUMB
        EXPORT  Start

Start

; initialize Port D
; enable digital I/O, ensure alt. functions off

; activate clock for Port D
                LDR R1, =SYSCTL_RCGCGPIO_R          ; R1 = address of SYSCTL_RCGCGPIO_R
                LDR R0, [R1]                        ;
                ORR R0, R0, #0x08                   ; set bit 3 to turn on clock for GPIOD
                STR R0, [R1]
                NOP                                 ; allow time for clock to finish
                NOP
                NOP

; no need to unlock Port D bits
; disable analog mode
                LDR R1, =GPIO_PORTD_AMSEL_R
                LDR R0, [R1]
                BIC R0, R0, #0x09             ; Clear bit 0 and 3 to disable analog function
                STR R0, [R1]

; configure as GPIO
                LDR R1, =GPIO_PORTD_PCTL_R
                LDR R0, [R1]
                BIC R0, R0,#0x0000000F              ; clear PortA bit 0
                BIC R0, R0,#0X0000F000              ; clear PortA bit 3
                STR R0, [R1]

; set direction register
                LDR R1, =GPIO_PORTD_DIR_R
                LDR R0, [R1]
                BIC R0, R0, #0x01             ; set PortD bit 0 input
                ORR R0, R0, #0x08             ; set PortD bit 3 output (0: input, 1: output)
                STR R0, [R1]

; disable alternate function
                LDR R1, =GPIO_PORTD_AFSEL_R
                LDR R0, [R1]
```

```
                BIC R0, R0, #0x09                       ; disable alternate function on bit 0 and 3
                STR R0, [R1]

; pull-up resistors on switch pins
                LDR R1, =GPIO_PORTD_PUR_R               ; R1 = address of GPIO_PORTD_PUR_R
                LDR R0, [R1]                            ;
                ORR R0, R0, #0x01                       ; enable pull-up on PortD bit 0
                STR R0, [R1]

; enable digital port
                LDR R1, =GPIO_PORTD_DEN_R
                LDR R0, [R1]
                ORR R0, R0, #0x09                       ; enable digital I/O on bit 0 and 3
                STR R0, [R1]

                LDR R1, =PD

Again1
                MOV R0, #0
                STR R0, [R1]                            ; "off" buzzer
                LDR R2, [R1]                            ; check switch, PortD bit 0 status
                TST R2, #1                              ;
                BNE Again1                      ; perform a bitwise AND operation and test again if
                                                ; switch is not pressed
                MOV R0, #0x08                   ; when switch is pressed, set PortD bit 3 "high" to
                                                ; turn on buzzer
                STR R0, [R1]                    ;
Again2
                LDR R2, [R1]                            ; check switch
                TST R2, #1                      ; perform a bitwise AND operation and test again if
                                                ; switch is not released
                BEQ Again2
                B Again1

                ALIGN                                   ; make sure the end of this section is
aligned
                END                                     ; end of file
```

- Load and run the program.
- Make sure you understand the following points:
  - (i) How to read the status of a push button switch and turn on a buzzer via GPIO Port D
  - (ii) Branching and looping

Exercise

Combine all three programs into one that will use:

- Port A to read in the status of a dip switch.
- Port B to output a word specified by the dip switch.
- Port D to sound a buzzer when a push button is activated.

List of Figures

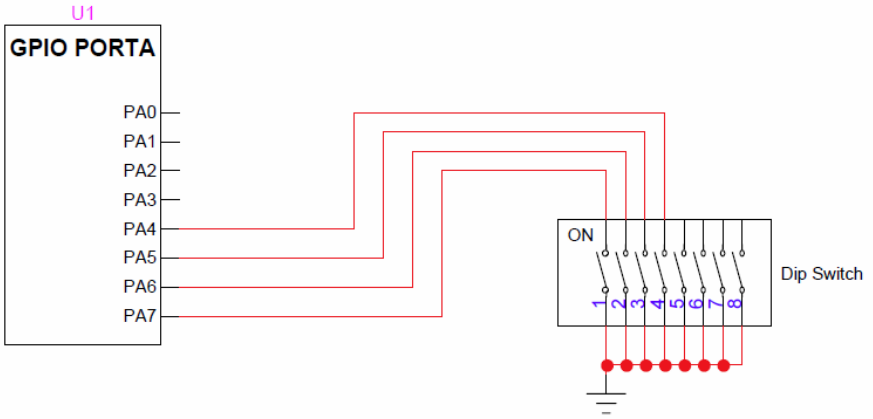


Fig. 1

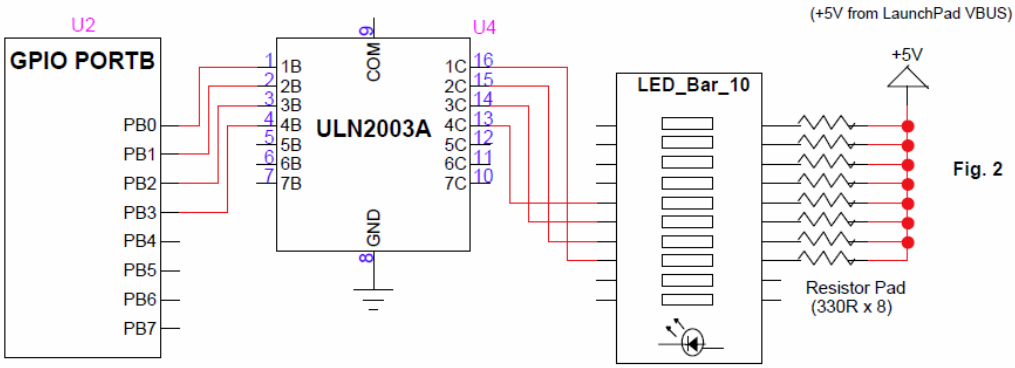


Fig. 2

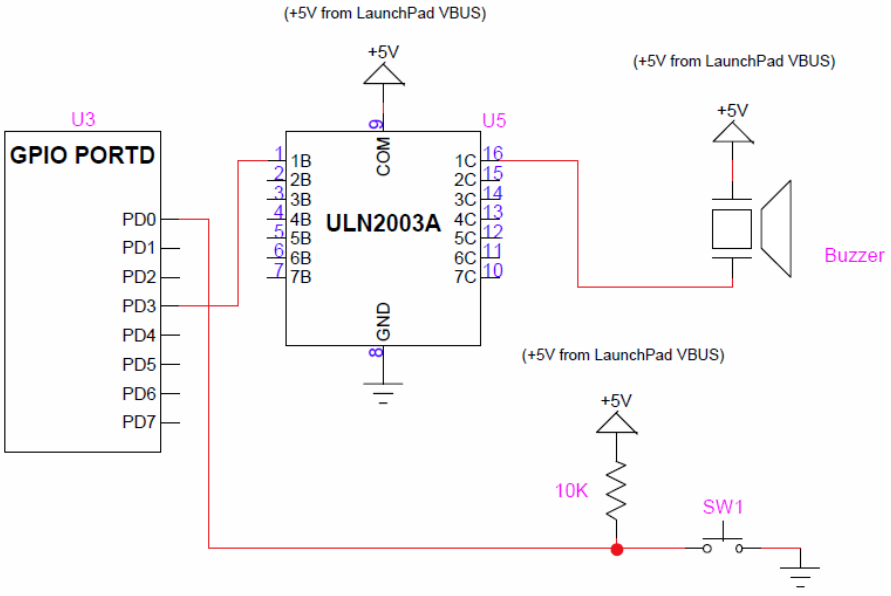


Fig. 3

Schematic (Lab 1)